predict
prioritise
prevent

TRESPASS

Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security

Deliverable D3.2.1

Extraction methods for stochastic models

Project:	TRE$_S$PASS
Project Number:	ICT-318003
Deliverable:	D3.2.1
Title:	Extraction methods for stochastic models
Version:	1.0
Confidentiality:	Public
Editor:	R. Kumar
Cont. Authors:	R.Kumar, A. Rensink
Date:	2015-10-30

Part of the Seventh Framework Programme
Funded by the EC-DG CONNECT

Members of the TRESPASS Consortium

1.	University of Twente	UT	The Netherlands
2.	Technical University of Denmark	DTU	Denmark
3.	Cybernetica	CYB	Estonia
4.	GMV Portugal	GMVP	Portugal
5.	GMV Spain	GMVS	Spain
6.	Royal Holloway University of London	RHUL	United Kingdom
7.	itrust consulting	ITR	Luxembourg
8.	Goethe University Frankfurt	GUF	Germany
9.	IBM Research	IBM	Switzerland
10.	Delft University of Technology	TUD	The Netherlands
11.	Hamburg University of Technology	TUHH	Germany
12.	University of Luxembourg	UL	Luxembourg
13.	Aalborg University	AAU	Denmark
14.	Consult Hyperion	CHYP	United Kingdom
15.	BizzDesign	BD	The Netherlands
16.	Deloitte	DELO	The Netherlands
17.	Lust	LUST	The Netherlands

Document History

Authors		
Partner	Name	Chapters
UT	R.Kumar	1, 2, 3, 4, 5
UT	A. Rensink	1, 2, 3, 4, 5

Quality assurance		
Role	Name	Date
Editor	Rajesh Kumar	2015-10-30
Reviewer	Michael Nidd	2015-10-15
Reviewer	Dieter Gollmann	2015-10-15
Task leader	Jaco van de Pol	2015-10-30
WP leader	Jaco van de Pol	2015-10-30
Coordinator	Pieter Hartel	2015-10-30

Circulation	
Recipient	Date of submission
Project Partners	2015-10-30
European Commission	2015-10-30

Acknowledgement: *The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TRESPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.*

Contents

List of Figures iv

List of Tables v

Management Summary vii

1. Introduction 1
1.1. Work package goals . . . 1
1.2. Context . . . 1
1.3. Goals of the deliverable . . . 2
1.4. Document structure . . . 3
1.5. Foreground and background . . . 4

2. Quantitative modelling: from system models to analysis models 5
2.1. The analysis process . . . 6

3. Attack trees 8

4. Extraction methods 10
4.1. Extraction of Markov Automata . . . 11
4.1.1. Motivation . . . 11
4.1.2. The model . . . 11
4.1.3. Semantics of the model . . . 12
4.1.4. Example: Attack of a password protected file . . . 15
4.2. Extraction of Weighted Timed Automata . . . 17
4.2.1. Motivation . . . 17
4.2.2. The model . . . 17
4.2.3. Example: Forestalling release of software . . . 20
4.3. Extraction of Interactive Markov Chains . . . 22
4.3.1. Motivation . . . 22
4.3.2. Semantics of the model . . . 23
4.3.3. Compositional aggregation . . . 24
4.3.4. Example: IPTV Case . . . 25

5. Conclusion 28

References 29

A. Project Summary **33**
A.1. Case Studies 34
A.2. Overview of TRESPASS Integration 35

List of Figures

1.1. WP3 integration within TRESPASS. 2
1.2. Different formalisms being developed within TRESPASS. Here, I/O IMC refers to Input/Output interactive Markov chains, MA refers to Markov automata and TA refers to Timed automata. Attack DSL refers to the attack domain specific language. 3

2.1. WP3 visualization of Abstraction Layers within TRESPASS 5

3.1. Example of attack tree for forestalling release of software 9

4.1. Representation of a BAS as MA. 12
4.2. Representation of parallel and sequential AND-gates as MAs. 13
4.3. Representation of parallel and sequential OR-gates as MAs. 14
4.4. Graphical overview of compositional aggregation for AT models. 15
4.5. Dynamic Attack Tree model: attack on password protected file. 15
4.6. Probability of success for case: Password protected file. 16
4.7. Sensitivity analysis for case: Password protected file 16
4.8. PTA for a basic attack step. Here v is a unique identifier for the BAS, x is a clock to track how long the BAS has been in progress, k is the number of costs to track, T_min and T_max and the minimum and maximum times, $costs$ is an array keeping track of all accumulated costs, and c_fix and c_var are arrays of the fixed and variable costs. The notation $0..k$ is used to indicate that the action is performed for all elements of the arrays. 18
4.9. PTA for parallel AND gate of node v, when $\mathrm{child}(v) = c_1 c_2$. 19
4.10. PTA for sequential AND gate of node v, when $\mathrm{child}(v) = c_1 c_2$. 19
4.11. Automaton for the attack goal Top 19
4.12. Optimal Resources (Time/ Cost) for Generic Attacker/ Software Engineer . . 22
4.13. Pareto curve of attack tree in Figure 3.1 22
4.14. Representation of BAS as IMC. 23
4.15. IMC representation of a AND-gate with children A and B. 23
4.16. IMC representation of a OR-gate with children A and B. 23
4.17. An interactive Markov chain with 4 states. 24
4.18. The probability that the attacker successfully attacks the IPTV system as a function over time. 25
4.19. The minimized IMC of the attack tree. 26

A.1. Legend for the Integration diagram in Figure A.2. 36
A.2. Integration diagram for the TRESPASS project. 37

List of Tables

4.1. Lower-level formalisms used for stochastic analysis after extraction from attack trees 10
4.2. Values used for annotating leaves of the attack tree as in figure 3.1 21

Management Summary

Key takeaways:

- We distinguish three levels of modelling: architectural (the socio-technical model), domain-sprecific (attack trees and variations thereof) and stochastic models (a large variety). This deliverable addresses the extraction of stochastic models from attack trees.
- Three different extraction methods are presented: to Interactive Markov Chains (reported before and included here for the sake of completeness), to Markov Automata, and to Weighted Timed Automata.
- This deliverable should be read in close conjunction with (The TRESPASS Project, D3.3.2, 2015), which details how the generated stochastic models are then analyzed. In practice, the properties to be analyzed drive the extraction process.

This is the first public deliverable of task T3.2 of the TRESPASS project. Within WP3, on "Quantitative Analysis Tools", this task is concerned with an intermediate step, which needs to be put into context to be properly understood.

The overall method to analyse socio-technical models has been cut up into several steps: first, from the socio-technical model (also called the "architectural model" in this deliverable), one generates "domain-specific" attack models referred as "attack DSL"; second, these are then further transformed into low-level stochastic models; third, the actual analysis is carried out on that low level; fourth and finally, analysis results are traced back to the original attack and socio-technical levels. This deliverable addresses the second step; this is what is called *extraction of stochastic models*.

Three further points should be noted:

- Extraction of stochastic models typically goes hand in hand with the third step, the analysis of the extracted models. In fact, the analysis questions in step 3 determine what needs to be extracted in step 2. Thus, this deliverable should be read in conjunction with the (simultaneously appearing) (The TRESPASS Project, D3.3.2, 2015), which addresses the analysis methods.
- There is an alternative route to that described above, which bypasses the extraction stage and goes from the architectural level directly to the low-level stochastic model level; namely, Timed Automata-based analysis of attacks. Since this does not involve extraction, this deliverable does not further address that route.

- Some of the analysis methods reported in (The TRESPASS Project, D3.3.2, 2015) directly work on the domain-specific attack trees, without requiring a further extraction step to any dedicated stochastic model. These, too, are ignored in this deliverable.

The extraction methods reported in this deliverable are:

- Extraction to Interactive Markov Chains. These models distinguish between non-deterministic and stochastic behaviour, and hence can answer questions about attacks combining one-time voluntary choices and steps whose probability of success is a function of time.
- Extraction to Markov Automata. These models extend IMCs with probabilistic behaviour, involving discrete probabilities rather than continuously evolving ones.
- Extraction to Weighted Timed Automata. These models include a notion of cost, which allows one to express and reason about the expected gain or required attacker budget.

Each of the extraction methods is described in some detail, referring to published or submitted papers for the full story, and is accompanied by a small illustrative example.

1. Introduction

1.1. Work package goals

WP3 develops the quantitative analysis techniques and tools for TRESPASS. Thus, it quantifies the attack scenarios derived from socio-technical models — see (The TRESPASS Project, D1.1.1, 2013) — and enables to answer questions such as: "What is the probability of attack within t days?", "What is the cost optimal path for an attacker given his skills and resources (budget and time constraints)?" and "Which countermeasures should a risk manager prioritize in his enterprise over the short term?". Hence, WP3 serves as a computational engine for the rest of project.

1.2. Context

The tasks in this WP are tightly coupled, either by requiring components from other work packages, or providing inputs to other work packages. Figure 1.2 provides a brief overview of WP3 and tasks as in TRESPASS.

Figure 1.1 taken from (The TRESPASS Project, D3.3.1, 2013), shows the close links between WP3 and WPs 1, 2 and 4. Data from WP2 (Data management process) is gathered to support model development in WP1 (Socio-technical security model specification). WP3 then analyses the model and associated data in order to identify attack scenarios and countermeasures and the results of this process may be visualised by WP4 (visualisation process and tools). The case studies provided by WP7 serve to validate the quality and applicability of the range of techniques developed in WP3. The set of algorithms and analysis techniques form a library of tools, to be included in the tool set produced by WP6.

Since there is no unique perfectly versatile model to represent all possible intelligent attacker behaviour, the partners in WP3 have explored different analysis techniques, summarised in Figure 1.2. An overview and comparison of these analysis techniques is given in (The TRESPASS Project, D3.3.2, 2015).

Focus

As detailed in The TRESPASS Project, D3.1.1 (2013), WP3 focusses on three main classes of quantities:

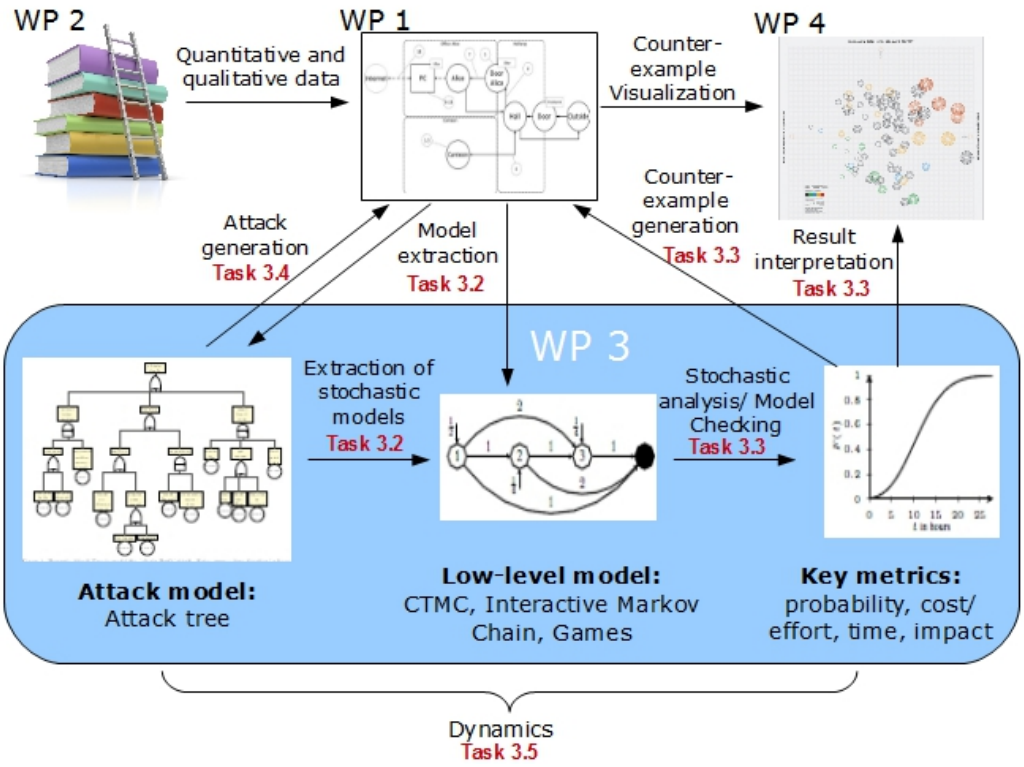


Figure 1.1.: WP3 integration within TRE$_S$PASS.

1. Probability, both discrete and continuous;
2. Time;
3. Effort, cost and impact.

The reason for this selection is that these quantities constitute key aspects in many of the socio-technical security problems modelled in WP1. In addition, a range of analysis techniques and tools is available to project partners for handling these quantities.

1.3. Goals of the deliverable

This deliverable focusses on Task T3.2 (cf. figure 1.2), which is the extraction of stochastic models from attack DSL, that are further analysed as part of other tasks in WP3.

The choice of extraction method is tightly coupled to the analysis techniques and quantitative metric of interest. Several general-purpose lower-level formalisms (I/O interactive Markov chains, Markov automata, Timed automata) are presented. While we reason about the time dependent attacks by deriving I/O interactive Markov chains and Markov automata from attack tree, we provide optimum attack values and paths by transforming attack trees into weighted timed automata.

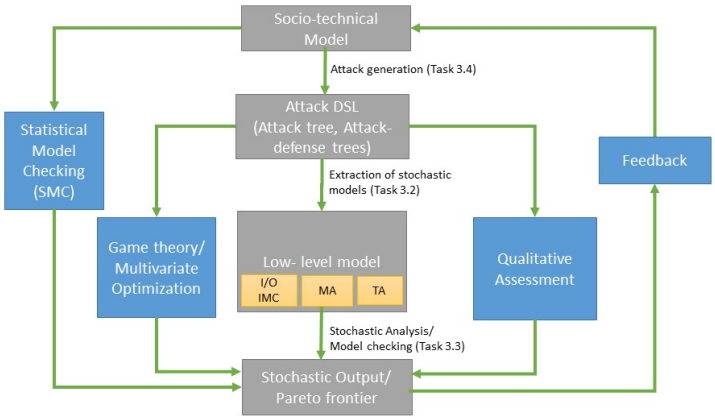


Figure 1.2.: Different formalisms being developed within TRE$_S$PASS. Here, I/O IMC refers to Input/Output interactive Markov chains, MA refers to Markov automata and TA refers to Timed automata. Attack DSL refers to the attack domain specific language.

All the extraction methods are validated through small case-studies either picked directly from the literature or provided by TRE$_S$PASS industrial partners. The aim of this deliverable is not to present analytical results, as that is done as part of task T3.3 in (The TRE$_S$PASS Project, D3.3.2, 2015).

1.4. Document structure

Appendix A provides the context for this deliverable in the TRE$_S$PASS project. It describes the overall summary of the project and the TRE$_S$PASS workflow. It provides an overview of conceptual system architecture, supported components and stages: data collection, model creation, analysis and visualisation which forms the basis of development of TRE$_S$-PASS workflow.

In chapter 2, we explain the need for an intermediate step of *extraction* from socio-technical model to stochastic analysis. To make the document self-contained, we explain the concepts of attack DSL and attacker profile in chapter 3. In chapters 4, we provide a family of quantitative automata (Markov automata, Priced timed automata and I/O Interactive Markov chains) and present the corresponding extraction methods. Each of the methods is accompanied by an example.

The current deliverable is a prototype deliverable; a screencast of the current tool implementation is provided on `https://vimeo.com/channels/trespass` (ATCalc). This will

help to provide a visually driven understanding of the analysis technique of which the extraction is a part.

1.5. Foreground and background

The current deliverable presents the state-of-the-art extraction techniques from attack DSL. Previous work on socio-technical model and on attack trees in the TRE$_S$PASS project are considered as background. The deliverable (The TRE$_S$PASS Project, D6.2.2, 2015) serves as the basis document for the requirements of the project and should be referred for understanding the overall goals of the project.

The foreground consists of a number of extraction techniques which serves as input for some analysis techniques developed in Task T3.3. One of the three extraction technique has been discussed earlier in (The TRE$_S$PASS Project, D3.3.2, 2015) thus, this deliverable has around 66 % new content.

2. Quantitative modelling: from system models to analysis models

To establish a common quantitative analysis framework for system security, WP3 distinguishes three levels in modelling, as shown in figure 2.1: the architectural level (UML, AADL, Archimate, E3value), the attack DSL (Attack trees (AT) and Attack-Defense trees (ADT)) and the analytical level (I/O interactive Markov chains (I/O-IMC), Markov automata (MA), Stochastic Games, Timed Automata). These three levels corresponds to three distinct trajectories in model development each of which can be used as starting point to run and derive analytical results. Their relative merits are:

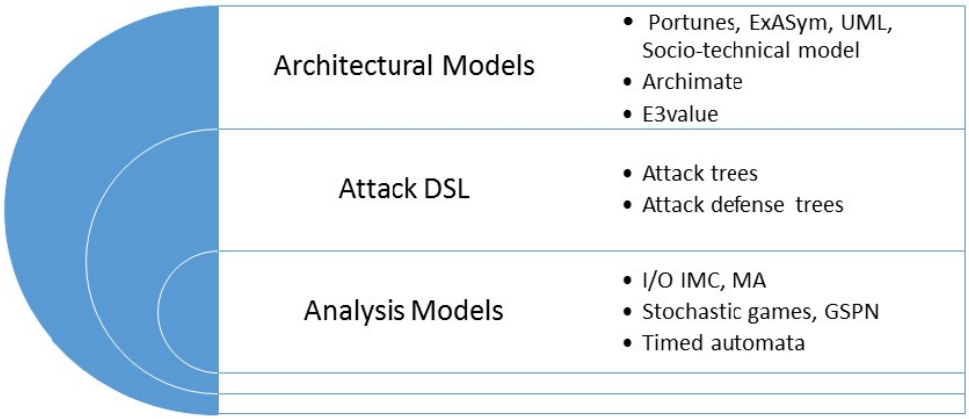


Figure 2.1.: WP3 visualization of Abstraction Layers within TRE$_S$PASS

Architectural modelling languages such as AADL (Rugina, Kanoun, & Kaâniche, 2007), Archimate (Lankhorst, Proper, & Jonkers, 2010), CORAS (Fredriksen et al., 2002) and UML are widely used in modelling dynamic systems for code generation, system validation and verification. They have been extended with security extensions such as UMLsec, Secure UML (a UML profile) (Jürjens, 2005; Basin, Doser, & Lodderstedt, 2006). They are quite expressive, easy to model in and thus save a lot of modelling costs and efforts. However, their semantics remain ambiguous and the models themselves tend to be restrictive with limited constructs, hence their behaviour need to be supported with the lower-level formalisms. It is easy to build the model here as standard model pattern libraries are readily available. In TRE$_S$-PASS, we refer the socio-technical model as the architectural model, which is being developed as part of WP1.

Domain specific languages such as attack trees (Kordy, Piètre-Cambacédès, & Schweitzer, 2014) and fault trees (FAA, 2000) have intuitive graphical representations. These models traditionally tend to be static with restrictive syntactic constructs. Extensions of attack trees have been proposed, such as dynamic attack trees with additional gates of SAND (Sequential-AND) and SOR (Sequential-OR) over conventional AND and OR gates (Arnold, Guck, Kumar, & Stoelinga, 2015a; Jhawar, Kordy, Mauw, Radomirovic, & Trujillo-Rasua, 2015). The semantics of these extended versions of attack trees, modelling causal and temporal dependencies, are essentially expressed using variants of Markov chains (Lower order formalism). The DSLs are quite modular and simple to construct but have limited modelling power.

Lower-level analysis models provide a clear semantics to the behavioural properties of higher-level formalisms. Moreover, they are compositional. Compositionality means that a model can be built by dividing it into smaller sub- models that are easier to understand and analyse. Furthermore, such lower-level models are quite flexible in accommodating temporal and causal dependencies of the system attributes. Thus, they are quite useful to model the real-time continuous dynamics, capturing the system's non-deterministic behaviour.

Compositionality is important in socio-security scenarios as the considered scenarios are complex and constructing and analysing them directly usually suffers from state space explosion, necessitating abstractions and thus making models imprecise. Lower-level formalisms give the flexibility of transforming attacks into automata, which can then be used as subsystem in a larger system. This keeps the model modular, flexible and easy to extend (Boudali, Crouzen, & Stoelinga, 2007, 2010; Eisentraut, Hermanns, & Zhang, 2010a).

In order to systematically obtain the best features of the above discussed three levels in terms of modularity, clear semantics, expressiveness, and modelling effort, WP3 translates the socio-technical model to an attack tree which is subsequently iteratively reduced to a stochastic model (Boudali, Haverkort, Kuntz, & Stoelinga, 2007). This stochastic model is then used to obtain analysis metrics using static analysis/model checking.

2.1. The analysis process

Given the three-layered architecture and a particular choice of intermediate representation language ("Attack DSL"), we can naturally split the procedure into two subtasks: getting from the architectural, socio-technical model to the Attack DSL, termed "Attack generation", and from the Attack DSL to the desired stochastic model variant which can be further analysed and quantified, termed "Stochastic model extraction". These steps are further elaborated as:

Attack generation With the term "attack generation", WP3 members have a common understanding of deriving an attack DSL from the (architectural) socio-technical model.

Attack trees (cf. section 3) were chosen as attack DSL owning to their intuitive, hierarchical formal structure quite popular in risk assessment and WP3 partners experience in working with them. Several extensions of attack trees have been well studied in literature (Kordy et al., 2014) and in earlier deliverables (The TRESPASS Project, D3.4.1, 2014; The TRESPASS Project, D3.3.1, 2013) and used in the analysis in TRESPASS.

Extraction of stochastic models While attack trees are a classical way to model the attacks, they have limited expressiveness in modelling causal and time dependencies. Moreover, handling multiple attributes in the basic attack steps steps (i.e leaves in attack trees) is not very straightforward, unless we express the attack tree as a behavioural model (automaton). While the quantitative analysis in (LeMay, Ford, Keefe, Sanders, & Muehrcke, 2011a; Buckshaw, 2014) is based on simulations, in TRESPASS we chose model checking (Baier & Katoen, 2008) as the computational engine. The motive behind it, is to obtain precise assessment rather than an estimate value.

Model checking basically relies on a systematic state space exploration in verification of a property expressed in temporal logic. It is an established method in quantitative model based evaluation and is well-explored in the security domain (Sheyner, Haines, Jha, Lippmann, & Wing, 2002). In addition to checking whether the given property is satisfied or not, it also provides a trace (counter-example) if the property is *not* satisfied.

In this deliverable, given our desired security metric, we extract suitable automata (cf. section 4) from attack trees, which are further composed and subsequently analysed, e.g., by feeding into them into model checker. Several low-level formalisms such as Interactive Markov Chains, Markov Automata and Timed Automata are reported in this deliverable.

Stochastic analysis The resulting stochastic analysis models, such as attack trees either directly derived from socio-technical model or an automata of the kind discussed in the previous item, are further analysed using static analysis or model checking to answer the quantitative metric (The TRESPASS Project, D3.3.2, 2015) (for instance, probability or cost to reach the asset). All these analysis methods are studied as part of task T3.3 and reported in (The TRESPASS Project, D3.3.2, 2015) deliverable.

Those methods for which tool support has been developed in the TRESPASS project all hide the details of the underlying analysis models and techniques. In the end, they will be integrated and provided with a user interface (developed in WP6), thus abstracting the user away from the conceptual complexities.

3. Attack trees

For self-containedness of the current deliverable, we briefly describe our attack DSL (attack trees) and the concept of attacker profile (a persona who is motivated to execute an attack on the system).

An attack tree (AT) is a tree — or more precisely, a directed acyclic graph — that describes how combinations of attack steps can lead to a system breach. It consists of a *root*, representing the attacker's goal. The root is further refined into subgoals via *gates*, until the sub goals cannot be refined any further and the *basic attack steps (BASs)* constituting the leaves of the attack tree are reached. When subtrees are shared, ATs can be directed acyclic graphs, rather than trees.

Attack trees were chosen as formalism in WP3 to represent attacks because:

- They are an established formalism in both industry and academia to reason about risk assessment;
- Their hierarchical multi-step compact architecture is quite intuitive, even for non-security professionals;
- They offer flexibility in choice of degree of detail and level of decomposition;
- Partners in WP3 have long experience in working with this formalism.

Basic attack steps Basic attack steps (BASs) represent individual atomic steps within a composite attack, and appear as leaves of the AT. They can be decorated by a set of attributes characterizing attacker actions based on his possessed skill – such as time and cost to successfully execute the attack step and an eventual probability of success. At present, these values are based on cognitive brainstorming, historical data or data-logs.

Gates. Classically, an attack tree uses AND- and OR-gates to describe the conjunctive and disjunctive composition of their child nodes. That is, to succeed in an AND-gate, the attacker has to succeed in all of its child nodes, whereas the OR-gate requires the attacker to execute at least one child node successfully.

The SAND- and SOR-gate are the sequential versions of the AND- and OR-gates that model a temporal dependency between the child nodes. The SAND-gate models attacks that are to be conducted in a specific order: only after the primary attack step (the first child node of the gate from the left-hand side) is successful, the attacker will start with the next child. Similarly, a SOR-gate models that an attacker first executes the primary gate, and only if that fails, falls back to the next option.

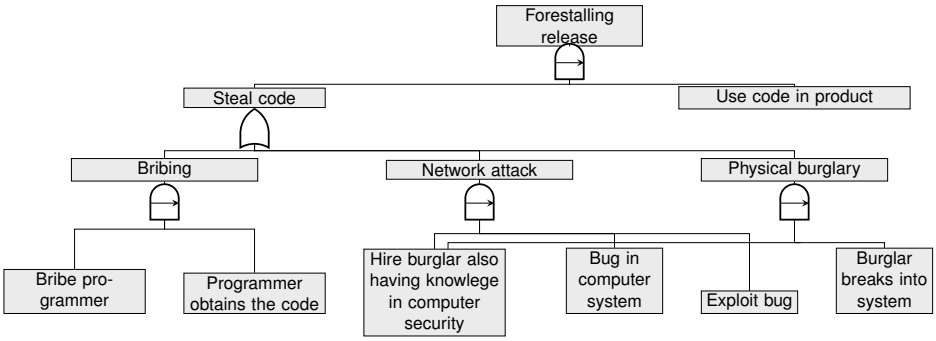


Figure 3.1.: Example of attack tree for forestalling release of software

Example of an attack tree. Fig. 3.1 shows an attack tree taken from (Buldas, Laud, Priisalu, Saarepera, & Willemson, 2006). The example has been modified by adding sequential temporal dependencies. Here, a competitor steals a piece of software code and then builds it into his own product, as modelled by the top-level SAND gate. The OR gate at node *Steal code* shows that the code can be stolen in three different ways: via *Bribing*, a *Network Attack* or *Physical burglary*. Bribing is modelled as a two-step sequential process of first successfully bribing a programmer and then obtaining the code, represented by a SAND-gate. Similarly, one can employ a burglar who has knowledge of computer security. This prerequisite, though possibly seeming vague, is added intentionally to show that our analysis tools can handle shared leaves/ sub-trees. Thus, this atomic step "Employ burglar having also having knowledge of computer security" can lead to two different attack paths modelled through a shared node. One in which the hired person finds a bug and exploits it to obtain the code via a network attack, and another one in which he is physically involved in a burglary after being hired to steal the code. This dependency is again modelled through a SAND gate.

Attacker profiles An attacker profile is a persona/institution, motivated to reach the asset (attack goal in attack tree) by executing basic attack steps in his intended order of preference. He starts with a limited budget, uses his skills and invests resources (time, cost) in executing basic attack steps and is eventually failed or successful with a certain probability. Personas offer a means of illustrating the different stakeholders and perspective in a risk scenario (The TRE$_S$PASS Project, D2.3.2, 2015; LeMay, Ford, Keefe, Sanders, & Muehrcke, 2011b; Buckshaw, 2014). It includes types of attacker (Malicious insider/ Disgruntled employee/ Competitor/ Irrational Individual), his capabilities, resources, initial knowledge of system and having an initial access to the system. Currently, we use predefined personae and their utilities based on extensive brainstorming for our case studies to demonstrate our proof of concept.

4. Extraction methods

The choice of stochastic model extracted for quantitative analysis is geared towards time – dynamic analysis. With the selection of appropriate formalism to model the behavioural aspects of the system model (here attack tree), our goal is to answer questions such as : " What is the probability of success of an attack within "t" units, given an attacker finite amount of resources to execute his basic attack steps (Time/ Cost) " or " Given an attacker profile (possessing a limited budget/skills to perform atomic attack steps), what is the cost optimal attack value and attack trace within certain 't' bound.

Thus, as a first step, we must consciously choose a suitable formalism to obtain the desired security metrics from an interconnected quantitative automata family. As we see in the table 4.1, each one of these formalism (Hartmanns & Hermanns, 2015) has its own distinctive modelling advantage with respect to feasibility, expressibility and compositionality and its ability to take non determinism, continuous dynamics and probabilistic choices into account.

Whereas Labelled transition systems (LTS) are quite expressive and can model non-deterministic behaviour, they cannot represent both continuous time dynamics and probabilistic choice. They are compositional in nature and thus, we can obtain a large LTS from multiple small LTS by synchronizing on action labels. Combining the two distinct formalism of LTS and CTMCs (Continuous time Markov chain) results into Interactive Markov chains (IMCs), which can be used to model both non-determinism and continuous dynamics. An extension of IMC is I/O-IMC, where the transition labels are separated into input and output actions. In contrast to CTMCs, IMCs can therefore synchronize on action labels and hence we can compose models. Adding discrete probabilities to IMCs result in another formalism – Markov Automata. These models can thus be used to model attacker behaviour of probability of success/ failure after an exponential time of execution of basic attack step. If we add continuous time in form of real-valued variables (called clocks) to LTSs, we obtain Timed automata (TA), which can be used to model non-deterministic, continuous dynamics of attackers.

Modelling formalism	Quantitative dimensions											
	Non-determinism			Probability distribution			Stochastic delays			Flows		
	None	discrete	continuous	Dirac	discrete	continuous	None	discrete	continuous	none	clocks	continuous
Labelled transition system	–	✓	–	✓	–	–	–	–	–	–	–	–
Continuous time Markov chains	–	–	–	✓	–	–	–	–	✓	–	–	–
Interactive Markov chains	–	✓	–	✓	–	–	–	–	✓	–	–	–
Markov automata	–	✓	–	–	✓	–	–	–	✓	–	–	–
Timed automata	–	–	✓	✓	–	–	–	–	✓	–	✓	–

Table 4.1.: Lower-level formalisms used for stochastic analysis after extraction from attack trees

4.1. Extraction of Markov Automata

4.1.1. Motivation

Measures on ATs We use Markov automata formalism (Guck, Hatefi, Hermanns, Katoen, & Timmer, 2013) to analyse ATs and provide insights on (1) static probabilistic questions such as probability to reach goal, (2) time-dependent questions such as probability of success as a function of time and (3) comprehensive questions from both models. Concretely, we can answer following security questions:

1. Given a probability distribution of attack execution time and eventual probability of success of atomic attack steps, what is the probability that system is compromised within time 't'?
2. How much time does an attacker need with a given success percentage to reach his goal?
3. Which basic attack step has the most impact in the attack tree? Putting it in context of temporal analysis, we can answer "The execution time of which BAS impacts the total execution time of whole attack scenario most significantly "

4.1.2. The model

Markov automata combine non deterministic, stochastic behaviour involving continuous time with probabilistic choices. Thus, they are a generalisation of interactive Markov chains (IMCs) and probabilistic automata (PAs). They are quite expressive and compositional. Here, we define two types of transitions: *Markovian* or *probabilistic*. A Markovian transition is labelled with a rate λ, indicating that this transition can be taken after an exponentially distributed delay with parameter λ. Thus, the probability to take the transition within time t equals $1 - e^{-\lambda t}$. Probabilistic transitions, labelled with Act leads to a probability distribution μ, that is, we move to the next state s with probability $\mu(s)$.

Input/ output Markov automata (I/O-MAs) extend Markov automata (MAs) (Eisentraut, Hermanns, & Zhang, 2010d, 2010b; Guck et al., 2013) by integrating the action behaviour of input/output automata (refer Figure 4.1 for an example) (Lynch & Tuttle, 1988) where:

- *Input actions* are the transitions denoted a? which are only possible if there is another I/O-MA that executes an output action a! and both can synchronize with each other. The action is thus controlled by the environment which can delay it;
- *Output actions* (denoted a!) cannot be delayed and are controlled by the respective I/O-MA. Transitions labelled with output actions have to be taken immediately;
- *Internal actions* (denoted a;) are controlled by the respective I/O-MA and cannot be delayed, similar to output actions. However, they are not visible to the environment.

I/O Markov automata are compositional, thus we can construct large automaton with several interacting automata composed in parallel. We denote this with $M_1||M_2$; the parallel composition of I/O-MAs M_1 and M_2. Thus, $M_c = M_1||M_2$ is again an I/O-MA (with the Cartesian product of M_1 and M_2 as state space) expressing the joint behaviour of its constituents where:

- if an action does not require synchronisation, then M_1 and M_2 evolve independently;
- if an action a on a probabilistic transition requires synchronisation, then both I/O-MAs must be able to perform the output action a! and corresponding input action a? simultaneously.

4.1.3. Semantics of the model

BAS model A BAS models a basic action which is executed by the attacker. This action is part of a more complex attack and interacts via gates with other BASs of this attack. We assign two attributes to our BAS (1) an exponential distribution λ, which is the time an attacker invests in the execution of the attack step (2) a eventual probability of success $P_success$. Although we have chosen exponential distributions, the analysis is also compatible with any phase distribution. λ signifies the amount of resources (here, time) that is expended during attack execution. This behaviour is described by the MA in Figure 4.1, which shows three different distributions which can govern the execution time: (1) an exponential distribution with parameter λ; (2) an $Erlang(3, \lambda)$ distribution; (3) and an arbitrary acyclic phase type distribution.

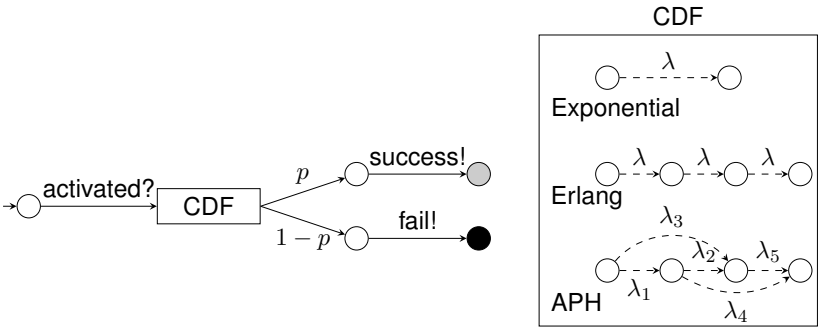


Figure 4.1.: Representation of a BAS as MA.

The MA is activated on receiving the input signal activated!, which enables the attacker to execute this particular step. Usually, the activation signal is sent by either the top-level node at system start or by a parent node. The execution of the attack step requires an amount of time that is governed by the associated CDF (cumulative distribution function). After execution, the attacker either succeeds with probability p or fails with probability $1-p$, which is denoted by branching into two different states. If the BAS is successful, it informs its parent nodes by sending a success! signal, otherwise, it sends a fail! signal.

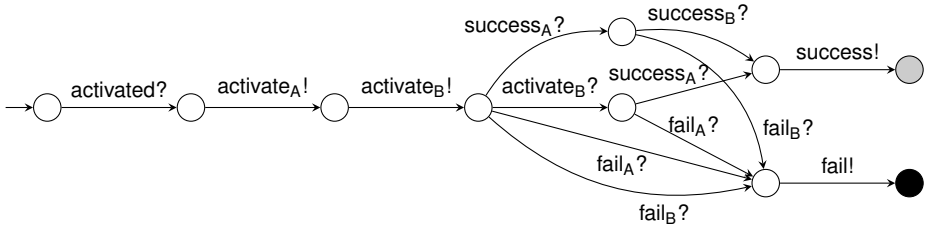
Parallel and sequential AND-gate model In an AND-gate, the children of parent nodes are all activated at the same time. If both the children succeed, the gate propagates a success signal to its parent node. The order of activation of child nodes does not matter.

To model temporal dependency between the child nodes, authors (Arnold, Hermanns, Pulungan, & Stoelinga, 2014) introduced the SAND gate where the execution from left to right imposes a restriction on the ordering of the attack event. Here, the right - most child is only activated if the left child is successful. This characterizes realistic behaviour, where an attacker need to successfully execute some atomic steps before proceeding with the next attack steps (For Example: Unlocking the door has three children: (1) successfully identifying the correct keys; (2) succesfully obtaining the correct keys; (3) and Succesfully unlocking the door.

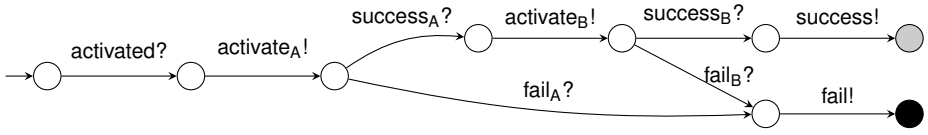
Modelling sequential behaviour using the SAND gate has the additional advantage of minimizing the number of states that needs to be enumerated by not activating the right children if the left-most child fails.

Semantics of AND-gate and SAND-gate model In Figure 4.2(a), the parallel AND gate waits for signals from A and B, which can be received in any order. If the attacker manages to penetrate both child nodes, the gate is successfully compromised and transmits a success! signal. If the attacker fails in the execution of at least one child node, the gate fails.

For a SAND-gate: The gate activates a child node only after it has received a success! signal from the previous child node. The corresponding model is shown in Figure 4.2(b). The attacker has to compromise A before he can execute B. If he does not succeed in A, the attack fails and the execution of B is not even started.



(a) MA representation of an AND-gate with children A and B.

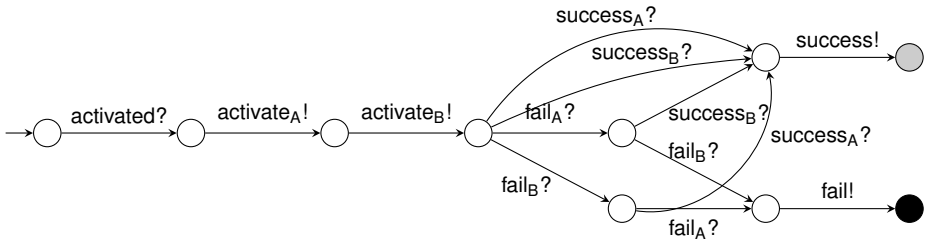


(b) MA representation of a SAND-gate with children A and B.

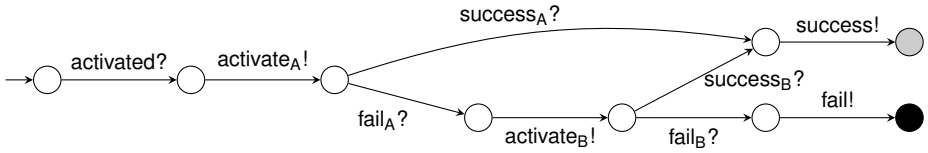
Figure 4.2.: Representation of parallel and sequential AND-gates as MAs.

Parallel and sequential OR-gate model The OR-gate represents a disjunctive composition of sub-trees or BASs. The attacker penetrates this gate as soon as one of the child nodes is attacked successfully. The parallel OR-gate is shown in Figure 4.3(a). On activation, the gate enables its child nodes A and B and the attacker executes both steps in parallel. As soon as he has succeeded in any one of the steps, the gate sends a success! signal to its parent nodes.

In contrast to that parallel OR-gate, the order of the execution of the child nodes matters in the sequential case, the SOR-gate. On activation, the gate enables the left-most child and waits for a signal. If the attacker succeeds in the execution of the child node, the gate is successfully penetrated. Otherwise, the attacker continues with the attack on the next child on the right to the previous BAS. The corresponding model is presented in Figure 4.3(b). Further information can be found in (Arnold, Guck, Kumar, & Stoelinga, 2015b).



(a) MA representation of an OR-gate with children A and B.



(b) MA representation of a SOR-gate with children A and B.

Figure 4.3.: Representation of parallel and sequential OR-gates as MAs.

Compositional aggregation The extraction of BAS into I/O Markov automata gives us the following freedom :

- I/O Markov automata are quite expressive. They allow modelling of temporal and causal dependency between attack steps. Also, they allow modelling of the probabilistic case of failure of an attack event after a mean time of execution (given by CDF).
- Instead of generating one aggregated model from an attack tree , we translate each BAS into respective I/O Markov automata which synchronise on their action labels. Thus, instead of composing the whole tree at once, we compose smaller sub-trees in a stepwise fashion and then minimise the state space after each composition using notions of strong, weak and branching bisimulation as studied in process algebra (Deng & Hennessy, 2013; Eisentraut, Hermanns, & Zhang, 2010c). Thus,

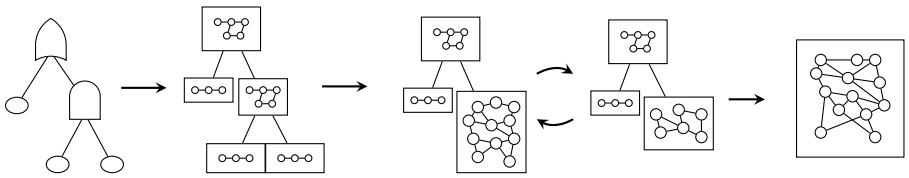


Figure 4.4.: Graphical overview of compositional aggregation for AT models.

we circumvent the omnipresent state space explosion by not constructing a large stochastic model in the first place. A graphical representation of the approach is shown in Figure 4.4.

4.1.4. Example: Attack of a password protected file

To make this document complete in itself, we summarize the stochastic analysis and results using the current approach of extraction of stochastic models (here Markov automata), refer (The TRESPASS Project, D3.3.2, 2015) for further more analysis method.

The following case study shown by attack tree, as in figure 4.5 is taken from (Piètre-Cambacédès & Bouissou, 2010). It is enriched by adding sequential gates to take care of temporal dependencies and has shared leaves, thus deviating from a pure tree formalism. Here, the goal is to obtain the password of a password protected file. There are two main ways to obtain it: (1) by cracking it through bruteforce, or (2) by trying to obtain the password.

We annotate each BAS with the mean time to execution (MTTE) defined by rate λ and a probability of success denoted by p. Those values are based on the intrinsic difficulty,

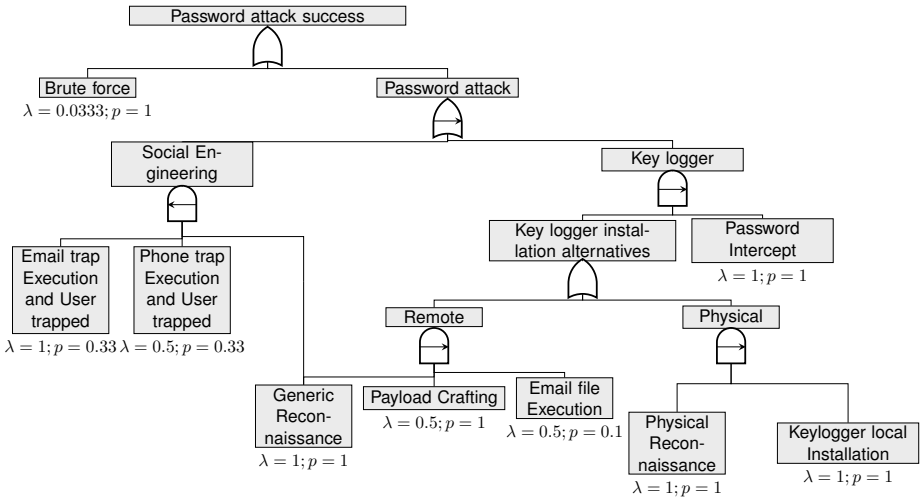


Figure 4.5.: Dynamic Attack Tree model: attack on password protected file.

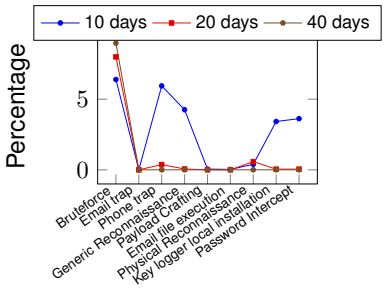


Figure 4.6.: Probability of success for case: Password protected file.

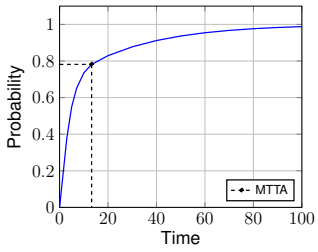


Figure 4.7.: Sensitivity analysis for case: Password protected file

available resources, estimated skills and the level of protection as provided in (Pièvre-Cambacédès & Bouissou, 2010).

Experimental results Fig. 4.6, depict the probability of a successful attack as a function over time. Our analysis results shows that the attacker succeeds with about probability of success of 61.9% after one week while the mean time of an successful attack is 13.2 days. To find, which attack step execution has the most serious impact in system security, we perform a sensitivity analysis (Ou & Dugan, 2000). Here, we run the analysis multiple times and in each run, we slightly change one of the BAS attributes while keeping the others fixed. Then, by ranking BAS in ascending order of percentage change in execution time, we can argue which BAS has the most impact on execution time of attack tree in temporal context. The added benefit of doing sensitivity analysis is that we can figure, how much the output is vulnerable to the input deviations.

Thus, by doing sensitivity analysis we can conclude that both the BASs *Phone trap execution with user trapped* and *Generic reconnaissance* remain equally important in the short term, i.e. 7-10 days. Figure 4.7 reveals that BAS– *Infection of Control PC*, *Intercept in/out signals* and *Collect data* have the greatest impact in the short and long term. To obtain these results, all experiments were computed on an Intel Xeon CPU E5335 at 2.00 GHz with 22 GB RAM under Linux. The average runtime for one experimental run for each case study is 76.57 sec for the password protected file.

Implication for security The timed dynamic analysis through Markov automata, provides a security practitioner of an estimate of how an attacker is expected to behave temporally – can the attacks be executed successfully in hours, days or rather months? In the context of enterprise security it equips system architects a tool to predict the likelihood of attack and thus make well informed decisions, by knowing which basic attack steps are more vulnerable than others and which countermeasures should be prioritized in short/ medium and long term.

4.2. Extraction of Weighted Timed Automata

4.2.1. Motivation

Measures on ATs. We use priced timed automata formalism to analyse ATs and answer questions such as : (1) Given an adversary persona, what is the minimal time / resources, or skill level needed to complete a successful attack? (2) Which attack path (a subtree of attack tree) will a rational attacker choose, if he wants to reach his goal minimizing his incurred cost?

Thus, our analysis framework can provide several useful metrics such as :

- *(Constrained) attack values*. For any of the attributes of atomic steps a_i (i.e time, cost), we can compute the minimum value along the tree. These values can be affected by constraints on other attributes. For instance, we can compute the minimum time needed to complete an attack within a maximum budget and skill level.
- *Pareto optimal curves.* For any pair of attributes, we can compute the minimum value needed of one attribute given a value of the other. By varying the bound of one attribute, we can generate curves indicating the relation between these minima. For instance, there is typically a trade-off between spending more time or more money; a Pareto curve shows for every budget how much time is needed for the attack.
- *Attack paths.* While computing the minimal attack value of an attribute that can reach to goal, we generate a concrete attack path showing the steps an attacker can take to perform the attack incurring as little of the attribute as possible. For instance, considering Figure 3.1, to reach the goal in the minimum time, we can obtain the attack trace which consists of *Hire a burglar*, *Burglar breaks into system* and *Use code in product*.
- *Ranking.* In addition to obtain the single minimum of an attribute and a corresponding attack path, we can enumerate further attacks in increasing value of the attribute. We can, for example, list the ten least costly attacks on a given system, or all attack paths that meet a given time constraint. For example, for the attack tree as shown in the figure 3.1, and the attributes annotated with the data as in Table 4.2, the optimal cost is 6000 units and the second best cost is 8500 units under no other attribute constraints.

4.2.2. The model

Timed automata (Alur & Dill, 1994), are the labelled transition system with real valued integers called *clocks* that synchronously increase over time. These clocks serve as *guards* over transitions serving as time bound after which an edge has to be necessarily taken or as invariants in locations, which implies that the location needs to be left after a residence time. Since they are extensions of LTSs, they are compositional in nature.

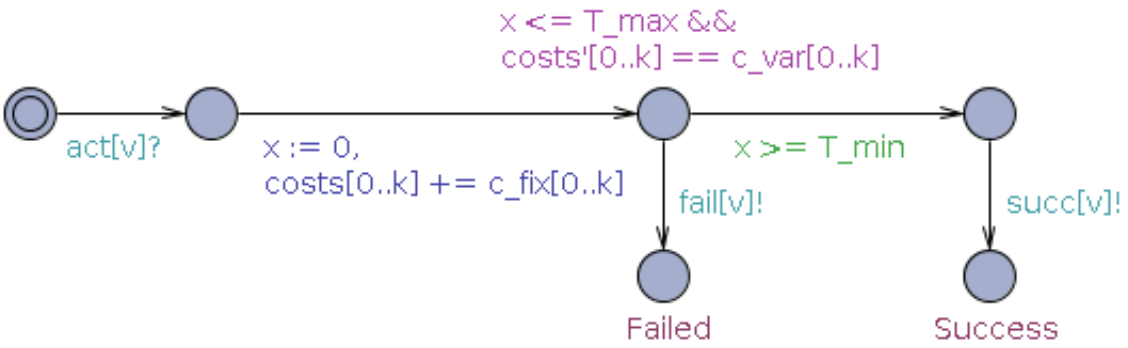


Figure 4.8.: PTA for a basic attack step. Here v is a unique identifier for the BAS, x is a clock to track how long the BAS has been in progress, k is the number of costs to track, T_min and T_max and the minimum and maximum times, $costs$ is an array keeping track of all accumulated costs, and c_fix and c_var are arrays of the fixed and variable costs. The notation $0..k$ is used to indicate that the action is performed for all elements of the arrays.

Priced/Weighted timed automata (PTA) (Behrmann, Larsen, & Rasmussen, 2005) extend timed automata, by adding costs to locations and actions transitions. Costs can either be accumulated in states, proportionally to the residence time, or by taking a transition. Both, TA and PTA can be analysed for a wide number of properties in model checking, including absence of deadlocks, safety, and liveness.

Technically, our framework is realized via Uppaal CORA (Brihaye, Bruyère, & Raskin, 2004): we translate each attack tree gate and leaf into a priced timed automata (PTA). Together they form a network of PTAs representing the entire attack tree. This modular approach yields a flexible framework that can easily be extended with future needs, such as countermeasures. We express our security queries in weighted CTL, and use the model checker Uppaal CORA to obtain the cost optimal traces that correspond to optimal attack paths in the AT. Optimality is defined in terms of a variable named cost. The optimal trace canvbe found by using the “Best trace” option in Uppaal CORA. More information can be found at (Kumar, Ruijters, & Stoelinga, 2015).

Semantics of BAS Fig 4.8 shows the semantics of the timed automata model derived for a basic attack step of the attack tree. This PTA models the attacker’s choice of whether and when to execute basic attack step, and tracks the time and costs expended to do so.

Semantics of Gates The PTAs for these gates begin by waiting for their activation signal, and then on obtaining *activate* signal proceeds with either activating all their own child nodes or only the left-most child . After this, they wait for success/failure signal of their child nodes. For an AND gate, receiving a failure signal always leads the gate to

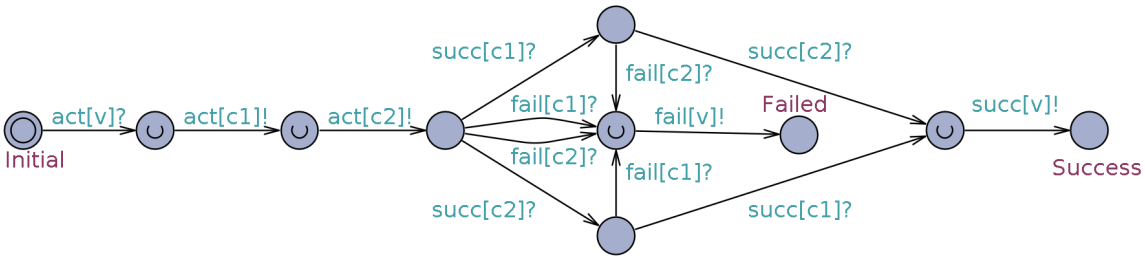


Figure 4.9.: PTA for parallel AND gate of node v, when $\mathrm{child}(v) = c_1 c_2$.

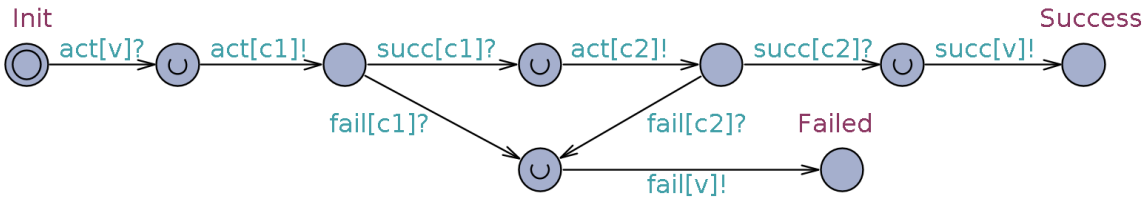


Figure 4.10.: PTA for sequential AND gate of node v, when $\mathrm{child}(v) = c_1 c_2$.

broadcast its own failure signal, since it is no longer possible for both children to succeed. Conversely, when an OR gate receives one success signal, it announces its own success. When both child nodes of an AND or OR gate have succeeded or failed, the gate also succeeds or fails respectively.

The sequential gates operate similarly, but they enforce an ordering on their children. First the left-most child is activated, and the gate waits for a signal from this child. In case of an SAND gate, success of the first child leads to an activation of the second child, and the success of this child cause the success of the gate. Failure of either child leads to failure of the gate, before even activating the second child. The behaviour of SOR is similar to sequential AND with success and failure signals swapped.

Combining the nodes. For an attack tree A, the PTA associated with A is obtained as the parallel composition of the PTAs for all the nodes, and an additional PTA A_{Top}. If we denote by $P(v, A)$ the PTA corresponding to node v of attack tree A, the total PTA consists of $P_A = P(v_1, A)||P(v_2, A)||\ldots||P(v_n, A)||A_{\mathsf{Top}}$.

The top-level gate Top is associated with a second PTA A_{Top}, shown in Figure 4.11 that initializes the attack by generating an activation signal for Top. Moreover, it has a clock x_{Top}

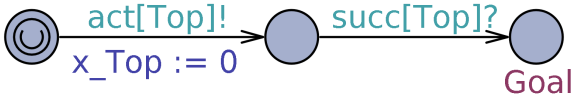


Figure 4.11.: Automaton for the attack goal Top

that tracks the global time, and observes successful completion of an attack via its input $succ[\text{Top}]?$. Thus, the location 'Goal' indicates that an attacker has reached the goal.

Analysis Techniques Optimal attack values can be obtained by repeatedly querying for the existence of trace reaching the attack goal with increasingly tight constraints. When the tightest possible bound has been obtained, this corresponds to an optimum. In addition, since a positive results for the query also produces a trace that satisfies it, this procedure also yields an optimal attack path. Note that the Uppaal program has a built-in method to find an optimum if only one cost needs to be tracked.

For example, to obtain the minimum time to succeed in the AT in Figure 4.5 given a cost limit of 1000 (assuming this is the only cost variable), we first query $\exists \diamond_{x_{\text{Top}} \geq 0, C \leq 1000} (P_{\text{Top}}.Goal)$ to obtain some successful attack and its corresponding time. Suppose this yields an attack that takes 10 days to complete, then we perform a new query $\exists \diamond_{x_{\text{Top}} < 10, C \leq 1000} (P_{\text{Top}}.Goal)$ to try to find a faster attack. If no such attack exists, we know that the minimal time to complete an attack given the budget is 10 days, and we have obtained an attack path that succeeds in this time and budget.

4.2.3. Example: Forestalling release of software

Description of Example This case is described through an attack tree as in figure. 3.1 in section 3. We annotate the BAS with attributes given in Table 4.2. An attacker skill is considered to be the precondition for beginning the attack and is defined in ordinal scale – low, medium and high. Based on the attacker persona and his skill level, we brainstorm the attacker attributes values – time bound required / cost required to execute the attack step.

Preconditions are basically boolean combinations over linear equations over attacker attributes (Skill, Costs, difficulty). In this way, an attack step that requires (at least) medium skill level is equipped with the enabling precondition $Skill \geq med$ (where med is a suitable constant); and an attack step that takes between 90 and 100 time units for medium-skilled attackers gets a success precondition $(Skill = med) \rightarrow (90 \leq T \leq 100)$. We choose *costs* specifically as – fixed and variable cost incurred by attacker and the cost incurred by company as damage done, refer Table 4.2. These costs are like effects on the attributes and are typically time dependent: the longer an attack takes, the higher the costs and damage. We assume that time dependence is linear, i.e., is incurred with a fixed rate v_i per time unit. Thus, the effect function is given by $Eff(a_i, (p_1, \ldots, p_n))(t) = f_i + v_i \cdot t$, where $f_i = f_i(p_1, \ldots p_n)$ and $v_i = v_i(p_1, \ldots p_n)$ are parameters that depend on the attribute values.

Here, we choose values just to illustrate our methodology. We believe that given at least rough bounds of realistic values, our analysis can provides insightful information about vulnerable paths and values relevant to risk managers.

We consider two attacker profiles. A generic attacker with a profit motive and a high risk appetite; and a malicious insider, a skilled software engineer, with better access, budget

and equipments, but a low risk appetite. Table 4.2 provides the parameters values used for the analysis.

Results In Figure 4.13, we see that both the attack values and the choice of attack path heavily depends on the attacker profile.Here, we see that both attack values and the choice of attack path heavily depend on the attacker profile. In contrast to the generic attacker whose cost optimal attack trace is to bribe a programmer, a better skilled software engineer exploits a bug in the computer system to steal the code. The minimum time required to accomplish the attack also heavily depends on which attack steps are executed and when. While a generic attacker takes 10 days to successfully execute the attack by physical burglary, a software engineer with insider benefits takes only 5 days to accomplish his goal. Also, there is an attack trace i.e *Hire a Burglar, Burglar breaks into system, Code is completed into product* which results in an *optimum Cost to company* as $500,000 irrespective of the considered attacker profiles. The analysis results are presented as a Pareto curve in Figure 4.13, where the generic attacker requires 10 days incurring a minimum cost of $9250 while a software engineer incurs a cost of $8500, but can complete the attack in 5 days.

Figure 4.12 provides a succinct representation of these different attack scenarios. We put the attacker objectives as vertices (i.e Minimum cost, Minimal time, Risk appetite, Cost to company) and the connecting lines are the attacker profiles (discussed is the case descriptions). The figure shows a trade-off among different attack values for the considered attacker profiles which an enterprise risk manager can use to effectively plan countermeasures.

Implication for security Priced timed automata provides a good representation of attacker model. By providing insights on system vulnerabilities through several dimensions, our analysis can be used by system designers to make well informed decisions in selection of countermeasures and in prioritizing their security investments.

	Attacker		Values		
BAS	Profile	Skill	Time (in days)	Cost (in US $)	Cost to company (in US $)
Bribe a programmer	Generic attacker	Low	15-20	1500 + 50t	500.000
	Generic attacker	Med	10-20	1000 + 150t	500.000
	Generic attacker	High	0-10	500	500.000
	Software Engineer	Any	0-5	5000 + 100t	500.000
Programmer obtains the code	Generic attacker	Any	5-15	1000 + 100t	1.000.000
	Software Engineer	Any	0-5	2000 + 50t	1.000.000
Hire burglar with knowledge of computer security	Any	Any	5-15	4000 + 50t	0
Bug in Computer system	Any	Low	15-20	1000 + 50t	0
	Any	Med	5-10	1000 + 50t	0
	Any	High	0-5	1000 + 50t	0
Person exploits the bug	Any	Any	0-5	1000 + 50t	1.000.000
Person breaks into the system	Any	Any	0-5	2000 + 100t	400.000
Code is completed into product	Any	Any	5-15	2000 + 50t	100.000

Table 4.2.: Values used for annotating leaves of the attack tree as in figure 3.1

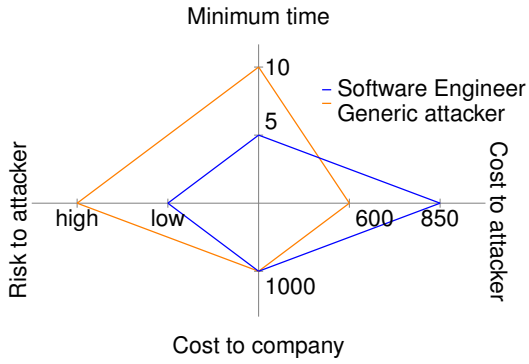


Figure 4.12.: Optimal Resources (Time/ Cost) for Generic Attacker/ Software Engineer .

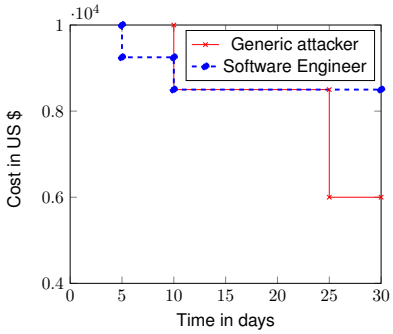


Figure 4.13.: Pareto curve of attack tree in Figure 3.1

4.3. Extraction of Interactive Markov Chains

Measures on ATs We use Interactive Markov chain (Hermanns, 2002) formalism to analyse ATs and provide insights on (1) static probabilistic questions such as probability to reach goal, (2) time-dependent questions such as probability of success as a function of time and (3) comprehensive questions from both models. Concretely, we can answer following security questions:

1. Given the probability distribution of attack execution time, what is the probability that system is compromised within time 't'?
2. How much time does an attacker need with a given success percentage to reach his goal?
3. Which basic attack step has the most impact in the attack tree? Putting it in context of temporal analysis, we can answer "The execution time of which BAS impacts the total execution time of whole attack scenario most significantly "

4.3.1. Motivation

Interactive Markov chains combines the stochastic behaviour with non deterministic choices but lack probabilistic choices. I/O interactive Markov chains extend interactive Markov chain by having the input and output action labels (Lynch & Tuttle, 1988), over which the transition systems can synchronize with each other. Thus, they are compositional in nature and are quite expressive formalism. Here, they can be used to model the attacker behaviour of eventually obtaining success after a given probabilistic distribution of time (CDF).

An example of Interactive Markov chain is given in Figure 4.17. The dashed lines represent Markovian transitions are taken only after an exponentially distributed time; given by λ – the rate of the distribution. They have basically have two transitions:

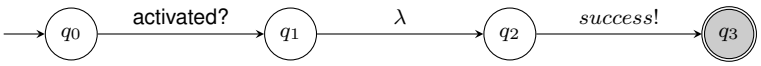


Figure 4.14.: Representation of BAS as IMC.

Figure 4.15.: IMC representation of a AND-gate with children A and B.

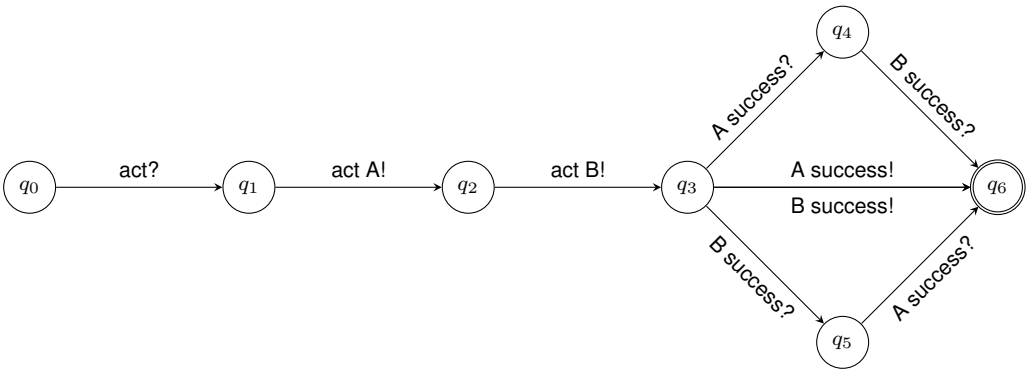


Figure 4.16.: IMC representation of a OR-gate with children A and B.

- Markovian transition: They model the stochastic behaviour, given by the rate of the probability distribution and is labelled with λ.
- Action transition: They are either labelled with action labels $Activated$ or an internal action τ. Input Actions labelled with $Activated?$ can synchronize with output actions labelled with action labels $Activated!$ and are executed in zero time (as in the CSP approach in process algebra) (Hoare, 1985).

When both Markovian and Actions transition are enabled, it is governed by maximal progress – i.e internal transitions cannot be delayed over Markovian transitions, though external transitions can be infinitely delayed.

4.3.2. Semantics of the model

BAS model A BAS is a basic step of an attacker or defender which interacts with a gate. We assign an attributes to our BAS– an exponential distribution λ, which is the time an attacker invests in the execution of the attack step. The fact, that we have used

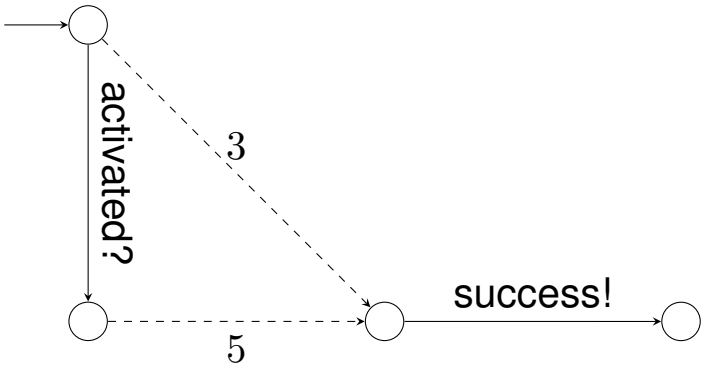


Figure 4.17.: An interactive Markov chain with 4 states.

exponential distribution is justified as it is the "most random" of all given distributions with a given mean, in the sense it has maximal entropy (Arnold et al., 2014). Although we have chosen exponential distributions, the analysis is also compatible with any phase distribution. Here, λ signifies the amount of resources (here: time, but can be cost as well) that is expended during attack execution. It is activated once it receives an *activation* signal from its parent node. After an exponential delay with rate λ the BAS propagates a *success* signal to its parent.

AND-gate and OR-gate model Figure 4.16 shows the semantics of BAS and gates. The activation signal propagates from top node to leaves (atomic attack steps) while the success signals propagate from bottom to top node.

- **AND:** An AND gate is a conjunction of events. Once it is activated by receiving an activation signal from its parent, it activates all its child nodes. The gate sends out a success signal only if all of its attached child nodes are successful.
- **OR:** An OR gate is a disjunction of events. Once it is activated by receiving an activation signal from its parent, it activates all its child nodes. The gate sends out a success signal if any of its attached child is successful.

4.3.3. Compositional aggregation

For our analysis, we first translate the atomic attack steps and the gates into equivalent IO-IMC. Instead of composing the BAS and the gates together all at once, we follow an iterative step by step process, to compose the adjacent I/O IMC based on their action transitions. Performing iteratively the process are then composed we obtain a single minimal IMC, as in Fig. 4.4. The resulting CTMC is analysed by the state-of-the-art model checkers MRMC (Katoen, Zapreev, Hahn, Hermanns, & Jansen, 2011), IMCA (Guck, Han, Katoen, & Neuhäußer, 2012) to obtain the probability of a successful attack over time.

4.3.4. Example: IPTV Case

The extraction of I/O IMCs and thereby stochastic analysis using IPTV case (The TRESPASS Project, D7.2.1, 2014) has already been reported as part of The TRESPASS Project, D3.3.1 (2013). We here reproduce the text to make this document complete in itself.

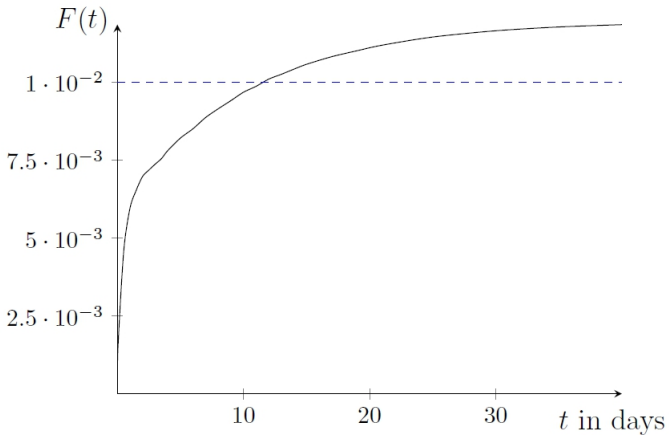


Figure 4.18.: The probability that the attacker successfully attacks the IPTV system as a function over time.

Description of Example This case was studied as part of the TREsPASS project (The TRESPASS Project, D7.1.2, 2015). It models a part of the security threats within a home-payment system for elderly people that allows care takers to make small payments on their behalf. The system can be configured via a TV set up box, hence its name IPTV. The process of deriving an IP/IMC from attack tree with the results have already been summarized in (The TRESPASS Project, D3.3.1, 2013). For the purpose of making this deliverable complete in itself, we provide here key results.

- The IPTV attack tree derived after extensive brain storming is highly complex with 346 nodes and a depth of up to 9 levels. The technique of compositional aggregation was used for aggressive state reduction techniques, resulting in 42 states iteratively reduced in 90 steps. This was used to find the timed-behaviour of the entire attack tree. The compressed model is displayed in Figure 4.19.
- The result of the analysis is a function which displays the probability of the attacker's success over time. Figure 4.18 shows that the probability of success increases rapidly in the course of the first day of the attack and reaches 0.7 %. This sharp initial increase can be explained by the fact that most attack steps can be executed in a very short time frame and only few have a long average execution time. The rate, at which the probability of success increases, falls steadily over the next days, reaching 1% after about 11 days. Further information can be found in (The TRESPASS Project, D3.3.1, 2013; Arnold et al., 2014, 2015b).

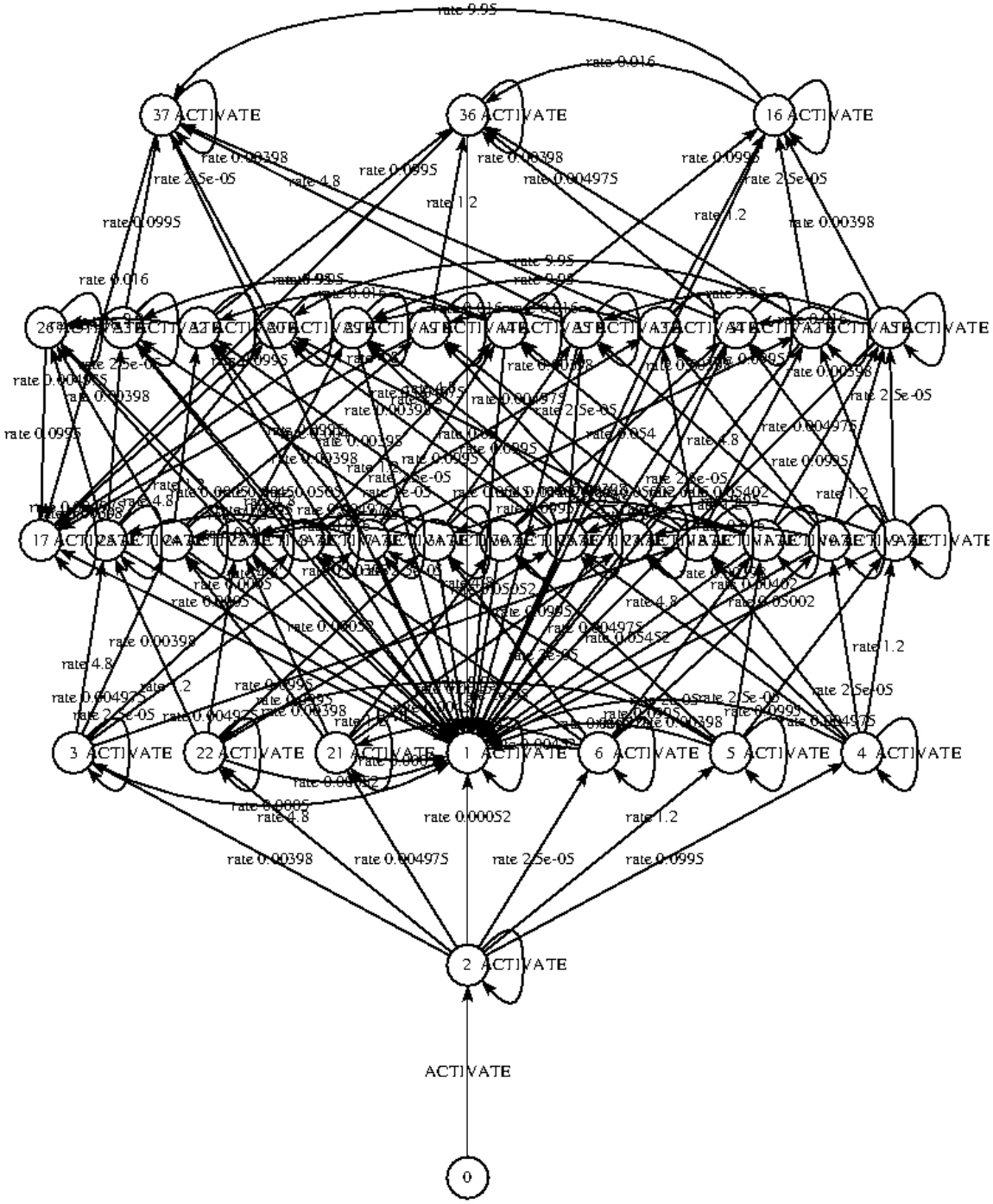


Figure 4.19.: The minimized IMC of the attack tree.

Implication for security Markov automata provides the enterprises with efficient techniques characterizing the evolution of attack. This can be used to make informed decisions on selection and prioritizing the security investments over countermeasures. By knowing

an estimate time bound over which an system is likely to be attacked, the system designers can know and fix the vulnerabilities in time.

5. Conclusion

This deliverable illustrates the work being done as part of Task T 3.2 in WP3 in accordance with the DoW.

The term ”Extraction of stochastic models” refers to an intermediate level between attack generation (via attack trees) and stochastic analysis. This is important as the models extracted from attack trees to a lower level dynamic formalism are semantically clean and are compositional in nature.

This deliverable takes several such extraction techniques – Markov automata, Interactive Markov chain, Priced timed automata; the choice of each formalism is tied to different analysis metrics studied as part of task T3.3 in (The TRESPASS Project, D3.3.2, 2015). Each of these formalism provides an insight of the attacker behaviour and characterizes system vulnerabilities, thus helping risk analysts to take well informed decisions in prioritizing their security investments over a subset of most vulnerable attack steps.

We have taken several case studies directly from literature in order to validate our methodology and provide an overview of security implications of the used formalisms.

References

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*(2), 183 – 235. doi: 10.1016/0304-3975(94)90010-8

Arnold, F., Guck, D., Kumar, R., & Stoelinga, M. (2015a). Sequential and parallel attack tree modelling. In *Computer safety, reliability, and security - SAFECOMP 2015 workshops, assure, decsos, isse, resa4ci, and sassur, delft, the netherlands, september 22, 2015, proceedings* (pp. 291–299). Retrieved from http://dx.doi.org/10.1007/978-3-319-24249-1_25 doi: 10.1007/978-3-319-24249-1_25

Arnold, F., Guck, D., Kumar, R., & Stoelinga, M. (2015b). Sequential and parallel attack tree modelling. In *Computer safety, reliability, and security - SAFECOMP 2015 workshops, assure, decsos, isse, resa4ci, and sassur, delft, the netherlands, september 22, 2015, proceedings* (pp. 291–299). Retrieved from http://dx.doi.org/10.1007/978-3-319-24249-1_25 doi: 10.1007/978-3-319-24249-1_25

Arnold, F., Hermanns, H., Pulungan, R., & Stoelinga, M. (2014). Time-dependent analysis of attacks. In M. Abadi & S. Kremer (Eds.), *Principles of security and trust* (Vol. 8414, p. 285-305). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-54792-8_16 doi: 10.1007/978-3-642-54792-8_16

Baier, C., & Katoen, J. (2008). *Principles of model checking*. MIT Press.

Basin, D., Doser, J., & Lodderstedt, T. (2006, January). Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, *15*(1), 39–91. Retrieved from http://doi.acm.org/10.1145/1125808.1125810 doi: 10.1145/1125808.1125810

Behrmann, G., Larsen, K. G., & Rasmussen, J. I. (2005). Priced timed automata: Algorithms and applications. *Formal Methods for Components and Objects*, *3657*, 162 – 182. doi: 10.1007/11561163_8

Boudali, H., Crouzen, P., & Stoelinga, M. (2007). A compositional semantics for dynamic fault trees in terms of interactive markov chains. In *Automated technology for verification and analysis* (Vol. 4762, pp. 441–456). Springer Verlag.

Boudali, H., Crouzen, P., & Stoelinga, M. (2010). A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *Dependable and Secure Computing, IEEE Transactions on*, *7*(2), 128–143.

Boudali, H., Haverkort, B. R., Kuntz, M., & Stoelinga, M. (2007). Best of three worlds: towards sound architectural dependability models.

Brihaye, T., Bruyère, V., & Raskin, J. F. (2004). Model-checking for weighted timed automata. *Proc. FORMATS-FTRTFT '04*, *3253*, 277–292.

Buckshaw, D. L. (2014). *Use of decision support techniques for information system risk management*. John Wiley Sons, Ltd.

Buldas, A., Laud, P., Priisalu, J., Saarepera, M., & Willemson, J. (2006). Rational choice of security measures via multi-parameter attack trees. In *Critical info. infrastructures*

security, first int. workshop, CRITIS, 2006 (pp. 235–248). doi: 10.1007/11962977_19

Deng, Y., & Hennessy, M. (2013). Compositional reasoning for weighted markov decision processes. *Science of Computer Programming*, *78*(12), 2537 - 2579. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167642313000440 (Special Section on International Software Product Line Conference 2010 and Fundamentals of Software Engineering (selected papers of {FSEN} 2011)) doi: http://dx.doi.org/10.1016/j.scico.2013.02.009

Eisentraut, C., Hermanns, H., & Zhang, L. (2010a). Concurrency and composition in a stochastic world. In *CONCUR 2010 - concurrency theory, 21th international conference, CONCUR 2010, paris, france, august 31-september 3, 2010. proceedings* (pp. 21–39). doi: 10.1007/978-3-642-15375-4_3

Eisentraut, C., Hermanns, H., & Zhang, L. (2010b). Concurrency and composition in a stochastic world. In *CONCUR 2010 - concurrency theory, 21th international conference, CONCUR 2010, paris, france, august 31-september 3, 2010. proceedings* (pp. 21–39). doi: 10.1007/978-3-642-15375-4_3

Eisentraut, C., Hermanns, H., & Zhang, L. (2010c). Concurrency and composition in a stochastic world. In *Concur* (Vol. 6269, p. 21-39). Springer.

Eisentraut, C., Hermanns, H., & Zhang, L. (2010d). On probabilistic automata in continuous time. In *Proceedings of the 25th annual IEEE symposium on logic in computer science, LICS 2010, 11-14 july 2010, edinburgh, united kingdom* (pp. 342–351). doi: 10.1109/LICS.2010.41

Fredriksen, R., Kristiansen, M., Gran, B., Stølen, K., Opperud, T., & Dimitrakos, T. (2002). The coras framework for a model-based risk management process. In S. Anderson, M. Felici, & S. Bologna (Eds.), *Computer safety, reliability and security* (Vol. 2434, p. 94-105). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-45732-1_11 doi: 10.1007/3-540-45732-1_11

Guck, D., Han, T., Katoen, J., & Neuhäußer, M. R. (2012). Quantitative timed analysis of interactive markov chains. In *NASA formal methods - 4th international symposium, NFM 2012, norfolk, va, usa, april 3-5, 2012. proceedings* (pp. 8–23). Retrieved from http://dx.doi.org/10.1007/978-3-642-28891-3_4 doi: 10.1007/978-3-642-28891-3_4

Guck, D., Hatefi, H., Hermanns, H., Katoen, J., & Timmer, M. (2013). Modelling, reduction and analysis of markov automata (extended version). *CoRR*, *abs/1305.7050*. Retrieved from http://arxiv.org/abs/1305.7050

Hartmanns, A., & Hermanns, H. (2015). In the quantitative automata zoo. *Science of Computer Programming*, -. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167642315002580 doi: http://dx.doi.org/10.1016/j.scico.2015.08.009

Hermanns, H. (2002). *Interactive markov chains: The quest for quantified quality* (Vol. 2428). Springer. Retrieved from http://dx.doi.org/10.1007/3-540-45804-2 doi: 10.1007/3-540-45804-2

Hoare, C. A. R. (1985). *Communicating sequential processes*. Prentice-Hall.

Jhawar, R., Kordy, B., Mauw, S., Radomirovic, S., & Trujillo-Rasua, R. (2015). Attack trees with sequential conjunction. In *ICT systems security and privacy protection - 30th IFIP TC 11 international conference, SEC 2015, hamburg, germany, may 26-28,*

2015, proceedings (pp. 339–353). Retrieved from http://dx.doi.org/10.1007/978-3-319-18467-8_23 doi: 10.1007/978-3-319-18467-8_23

Jürjens, J. (2005). *Secure systems development with UML*. Springer. Retrieved from http://dx.doi.org/10.1007/b137706 doi: 10.1007/b137706

Katoen, J., Zapreev, I. S., Hahn, E. M., Hermanns, H., & Jansen, D. N. (2011). The ins and outs of the probabilistic model checker MRMC. *Perform. Eval.*, *68*(2), 90–104. Retrieved from http://dx.doi.org/10.1016/j.peva.2010.04.001 doi: 10.1016/j.peva.2010.04.001

Kordy, B., Pièrre-Cambacédès, L., & Schweitzer, P. (2014). Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, *13-14*, 1–38. Retrieved from http://dx.doi.org/10.1016/j.cosrev.2014.07.001 doi: 10.1016/j.cosrev.2014.07.001

Kumar, R., Ruijters, E., & Stoelinga, M. (2015). Quantitative attack tree analysis via priced timed automata. In S. Sankaranarayanan & E. Vicario (Eds.), *Formal modeling and analysis of timed systems* (Vol. 9268, p. 156-171). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-319-22975-1_11 doi: 10.1007/978-3-319-22975-1_11

Lankhorst, M., Proper, H., & Jonkers, H. (2010, January). The anatomy of the archimate language. *Int. J. Inf. Syst. Model. Des.*, *1*(1), 1–32. Retrieved from http://dx.doi.org/10.4018/jismd.2010092301 doi: 10.4018/jismd.2010092301

LeMay, E., Ford, M., Keefe, K., Sanders, W., & Muehrcke, C. (2011b, sept.). Model-based security metrics using adversary view security evaluation (advise). In *Quantitative evaluation of systems (qest), 2011 eighth international conference on* (p. 191 -200). doi: 10.1109/QEST.2011.34

LeMay, E., Ford, M. D., Keefe, K., Sanders, W. H., & Muehrcke, C. (2011a). Model-based security metrics using adversary view security evaluation (ADVISE). In *Eighth international conference on quantitative evaluation of systems, QEST 2011, aachen, germany, 5-8 septemter, 2011* (pp. 191–200). Retrieved from http://dx.doi.org/10.1109/QEST.2011.34 doi: 10.1109/QEST.2011.34

Lynch, N., & Tuttle, M. (1988). An introduction to input/output automata.

Ou, Y., & Dugan, J. B. (2000, March). Sensitivity analysis of modules dynamic fault trees. In *Computer performance and dependability symposium, 2000. ipds 2000. proceedings. ieee international* (pp. 35–43).

Pièrre-Cambacédès, L., & Bouissou, M. (2010). Attack and defense modeling with BDMP. In *Proc. of the 5th int. conf. on mathematical methods, models and architectures for computer network security (mmm-acns)* (Vol. 6258, pp. 86–101). Springer Berlin Heidelberg.

Rugina, A., Kanoun, K., & Kaâniche, M. (2007). An architecture-based dependability modeling framework using AADL. *CoRR*, *abs/0704.0865*. Retrieved from http://arxiv.org/abs/0704.0865

Sheyner, O., Haines, J. W., Jha, S., Lippmann, R., & Wing, J. M. (2002). Automated generation and analysis of attack graphs. In *2002 IEEE symposium on security and privacy, berkeley, california, usa, may 12-15, 2002* (pp. 273–284). Retrieved from http://dx.doi.org/10.1109/SECPRI.2002.1004377 doi: 10.1109/SECPRI.2002.1004377

System safety handbook [Computer software manual]. (2000).

The TRESPASS Project, D1.1.1. (2013). *Initial specifications and requirements for socio-technical security models.* (Deliverable D1.1.1)

The TRESPASS Project, D2.3.2. (2015). *TRESPASS social data and policy extraction techniques.* (Deliverable D2.3.2)

The TRESPASS Project, D3.1.1. (2013). *Initial requirements for quantitative analysis tools.* (Deliverable D3.1.1)

The TRESPASS Project, D3.3.1. (2013). *First report on stochastic analysis methods.* (Deliverable D3.3.1)

The TRESPASS Project, D3.3.2. (2015). *TRESPASS methods for stochastic analysis.* (Deliverable D3.3.2)

The TRESPASS Project, D3.4.1. (2014). *Attack generation from socio-technical security models.* (Deliverable D3.4.1)

The TRESPASS Project, D6.2.2. (2015). *Final refinement of functional requirements.* (Deliverable D6.2.2)

The TRESPASS Project, D7.1.2. (2015). *Final requirements for implementation of case studies.* (Deliverable D7.1.2)

The TRESPASS Project, D7.2.1. (2014). *Results from case study a.* (Deliverable D7.2.1)

A. Project Summary

This chapter gives an overview of the TRE$_S$PASS project and its use cases. The section is shared by the public deliverables to provide the necessary background and to put the current deliverable in context.

Information security threats to organisations have changed completely over the last decade, due to the complexity and dynamic nature of infrastructures and attacks. Attacks like StuxNet involve technical and human factors, and they damage physical infrastructure. The recent attack on a German steel mill [1] was a combination of both targeted phishing emails and social engineering attacks. The phishing helped the hackers extract information they used to gain access to the plant's office network and then its production systems. As a result, the technical infrastructure of the mill suffered severe damage.

The attack on the German steel mill illustrates that we need to integrate the social and technical aspects of systems in assessing their security - and we need to do so today. Socio-technical systems pose new challenges by combining parts for which we often understand the security issues; the combined system is however much more complex due to interactions between these parts.

The main innovation of the TRE$_S$PASS project is the attack navigator, a tool and metaphor that enables defenders to predict and preventing attacks on socio-technical systems. The attack navigator supports current risk-assessment techniques with the TRE$_S$PASS process (developed in Work Package WP5), an analytical approach to identifying attacks and evaluating their impact.

The four main stages in the TRE$_S$PASS process are *data collection*, *modelling*, *analysis*, and *visualisation*. Data collection (WP2) is vital to understanding the nature of a scenario and providing input to subsequent tasks of modelling, analysis and visualisation. Within the project, the focus has been on collection and analysis of social, technical and physical data and the ways in which these relate to one another. Within each of these domains, different approaches have been taken to provide different viewpoints on the nature of the organisation being investigated.

The models (WP1) developed in TRE$_S$PASS can be adapted to the application scenario. We have developed physical modelling techniques in order to understand where further investigation may usefully be targeted. The TRE$_S$PASS model describes relevant aspects of the organisation and their connections. To explore contractual and commercial relationships, the e3value method has been adopted.

[1] BBC News, *Hack attack causes 'massive damage" at steel works*, http://www.bbc.com/news/technology-30575104, last visited October 31, 2015.

The analysis methods (WP3) developed in TRE$_S$PASS identify attacks in models and identify the most effective controls to prohibit these attacks. The analyses are supported by tools and together they provide the defender with a comprehensive understanding of properties attacks, *e.g.*, cost for the attacker, required skills, or required time.

The innovative visualisations (WP4) developed in TRE$_S$PASS focus focus particularly on visualising elements of the analysis, as this is key to the overall project goal of providing "decision support" to practitioners. However, visualisations contribute also to model development and data gathering.

Practitioners can access the TREsPASS toolkit via the attack navigator map interface, which provides an intuitive means of selecting appropriate tools (WP6) for data gathering, modelling, analysis and visualisation. These can be used, individually or in combination, to strengthen operational and strategic decision-making.

A.1. Case Studies

The TREsPASS process and tools are validated by means of case studies (WP7) in the area of cloud infrastructure, telecommunications infrastructure, ATM infrastructure, and an organisation processing privacy sensitive data.

A cloud infrastructure shares infrastructure within or across organisations, giving the cloud services provider and its employees full physical and logical access to all resources across the different consumers. In TRE$_S$PASS we formalise typical components in cloud infrastructures as well as human actors and their interrelationships, to identify their contribution to attacks on the organisation.

In telco infrastructure new products need to be launched under significant time pressure, often opening up loopholes for so-called knowledge insiders who know the market very well, trying to make as much monetary gain from the new products as possible. In TRE$_S$PASS we model both the infrastructure and contractual relationships to identify physical and monetary attacks.

The ATM infrastructure connects machines that are composed of a money safe and a computer that controls the ATM's devices. There are well protected ATMs installed inside bank branches, while others are deployed in the street and some are not even embedded in a wall. ATM attacks are common and include classic physical attacks and emerging digital attacks. In TRE$_S$PASS we model ATM installations, and identify attack likelihoods using geospatial data.

The organisation processing privacy sensitive data develops a system supporting primarily elderly and disabled people in performing online payments and managing their own money from their home. This case study involves from strictly technical security aspects, such as how information is protected while stored or transmitted, to socio-technical security aspects covering security issues arising from the use of and interaction with the technology. In TRE$_S$PASS we identify social-engineering and trust-based attacks on such systems.

A.2. Overview of TRESPASS Integration

The TRESPASS workflow involves several stages with various activities, some of which are optional. Figure A.2 shows the architecture diagram and Figure A.1 shows a visual description of the notation used. In practice, stages may not follow a linear order. For example, depending on the goal of the risk assessment, new data requests may be issued later in the process, or automatic updates of data may be supported.

The **Data collection stage** prepares for analysis and modelling steps, and may require the gathering of one or more of the following kinds of data.

Physical data collection provides knowledge about the physical layout of the organization including locations, buildings, rooms, doors, windows, etc.

Digital data collection gathers information about the organization's IT infrastructure.

Social data collection focuses on organisational and individual data, and results in actor profiles containing, *e.g.*, attributes of employees, stakeholders, or potential attackers.

Commercial data collection gathers information required for *e3fraud* analyses, which focus on potential fraud.

Stakeholder goal collection identifies assets and policies the protection of which is critical to one or more stakeholders.

The **model creation stage** handles the creation of the TRESPASS model and associated actor profiles. The e3value model creation process is complementary to the main TRESPASS model, for cases requiring a more specific financial focus:

TRESPASS model creation is a key activity result in a system model that can be further extended and analysed.

- **Components customization (optional)** takes place before or during the TRESPASS model creation to create specialized custom model components.
- **Attacker profile creation** creates the attacker profile that the TRESPASS model analysis should consider, based on ready-made attacker profiles.
- **Defender/target profile creation** creates similar profiles for the other actors in the model based on the social data gathered in the social data collection activity.

e3value model creation This interactive activity involves using the *e3value toolkit* [2] to create business value models. These models structure the commercial information gathered in the data collection stage in a formal way.

In the **analysis stage** different analyses are possible depending on the model chosen. The analysis of the TRESPASS model involves these steps:

1. In the **attacker profile selection**, the user selects the attacker profile to use in the analysis.

[2] http://e3value.few.vu.nl/tools/

2. The **attacker goals creation** provides the attack generation with the attacker goals. These can be derived by hand from the stakeholder goals or deduced automatically from the selected attacker profiles.
3. The **scenario selection** selects a scenario, consisting of a single pair of attacker and attacker goal, to run the TRESPASS analysis on.
4. To extend attack trees, **attack pattern creation and sharing** provides libraries with known attack steps. The attack tree generation can only reach a certain level of abstraction, which may not be sufficient for quantitative analyses.
5. **Attack generation** transforms the TRESPASS model to an attack tree.
6. **Attack tree annotation & augmentation** then extends the attack tree with attack patterns and decorates leaf nodes with parameter values from the data collection stage for quantitative analysis.
7. The **attack tree analyses** compute quantitative properties of attacks, *e.g.*, utility for the attacker or success probability of the attack.

The analysis of the **e3value model** is complementary to the core TRESPASS analysis and has only one step:

1. For the **fraud model generation**, the user needs select an attacker and an interval of expected occurrence rates of the commercial transactions specified by the e3value model. The e3fraud tool then identifies all possible violations of contracts, the loss for actors, and the delta in profit for the other actors.

The **visualisation stage** can be used continuously to provide practitioners with feedback regarding the results of their activities:

1. **Fraud model visualisation** shows the generated attacks as a ranked list of textual descriptions of the attack steps and displays charts showing the profitability for each actor.
2. **Attack tree visualisation** shows the intermediary attack trees.
3. **Attack tree analysis visualisation** visualises analysis results.

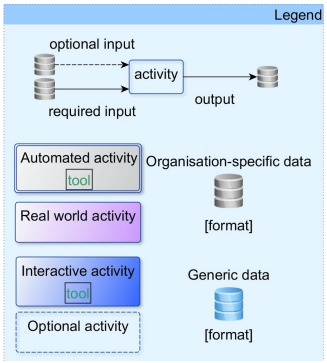


Figure A.1.: Legend for the Integration diagram in Figure A.2.

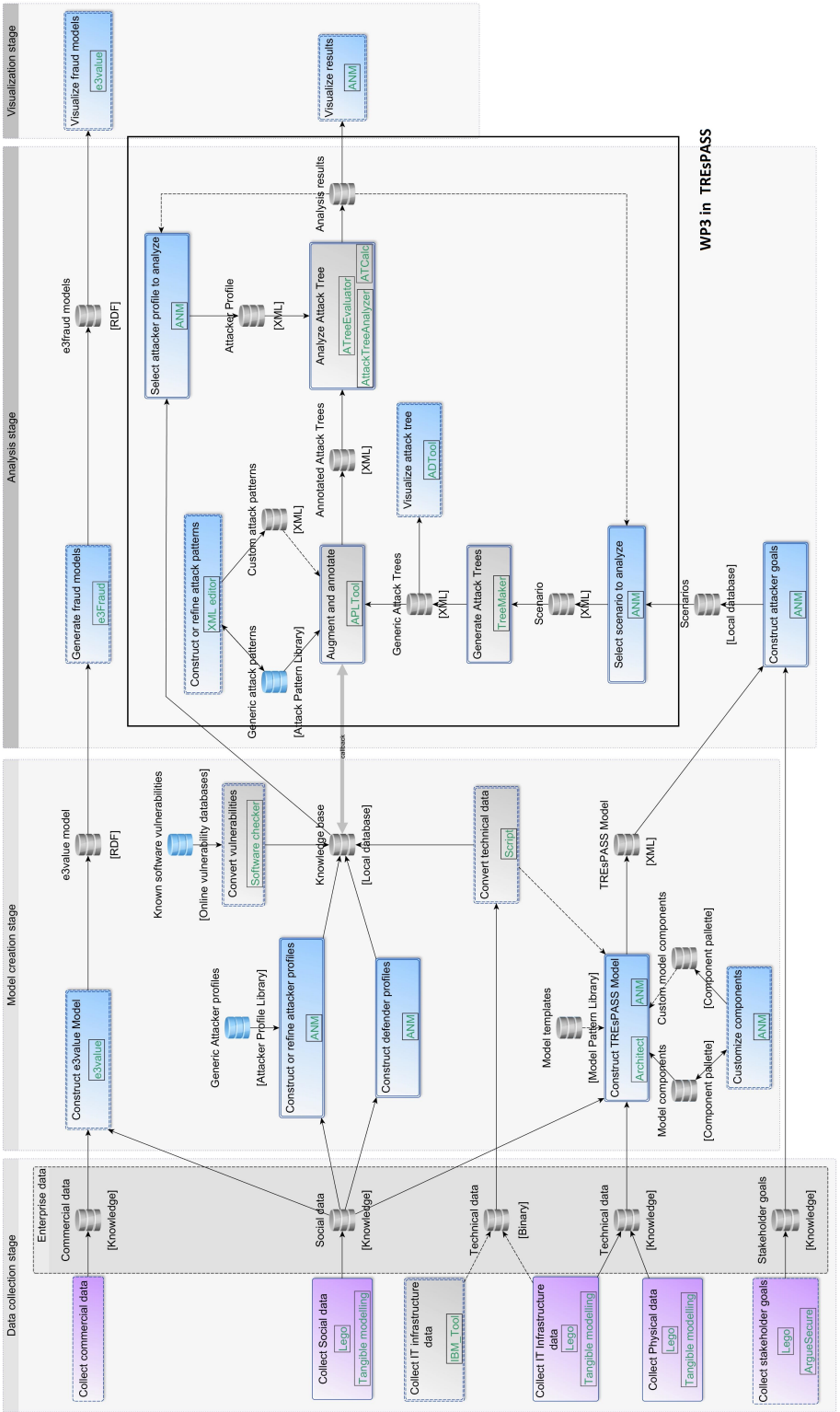


Figure A.2.: Integration diagram for the TRESPASS project.